INDEXING GEORGE WASHINGTON'S HANDWRITTEN MANUSCRIPTS

A STUDY OF WORD MATCHING TECHNIQUES

Shaun Kane
Andrew Lehman
Elizabeth Partridge

INDEXING GEORGE WASHINGTON'S HANDWRITTEN MANUSCRIPTS

A STUDY OF WORD MATCHING TECHNIQUES

ABSTRACT

There exists a large corpus of handwritten historical manuscripts that serve as a valuable resource to historians, but their inability to work with traditional information retrieval techniques can make them difficult to use. Before such techniques can be applied, some method to extract information about the textual content of a document must be applied. With typed material, existing optical character recognition (OCR) technology and known indexing techniques can work well. With handwriting, however, it is often difficult to segment characters from each other, but generally it is possible to clearly separate words. "Word spotting" is the process of creating a searchable text index based on the words contained within a document. This is accomplished by creating classes of word images, where each class consists of multiple instances of the same word. A human could then provide an ASCII equivalent for the classes. This paper discusses a number of techniques for normalizing for the quality of the word images and ranking their similarity. These algorithms were tested on a subset of the Library of Congress' collection of George Washington's handwritten manuscripts.

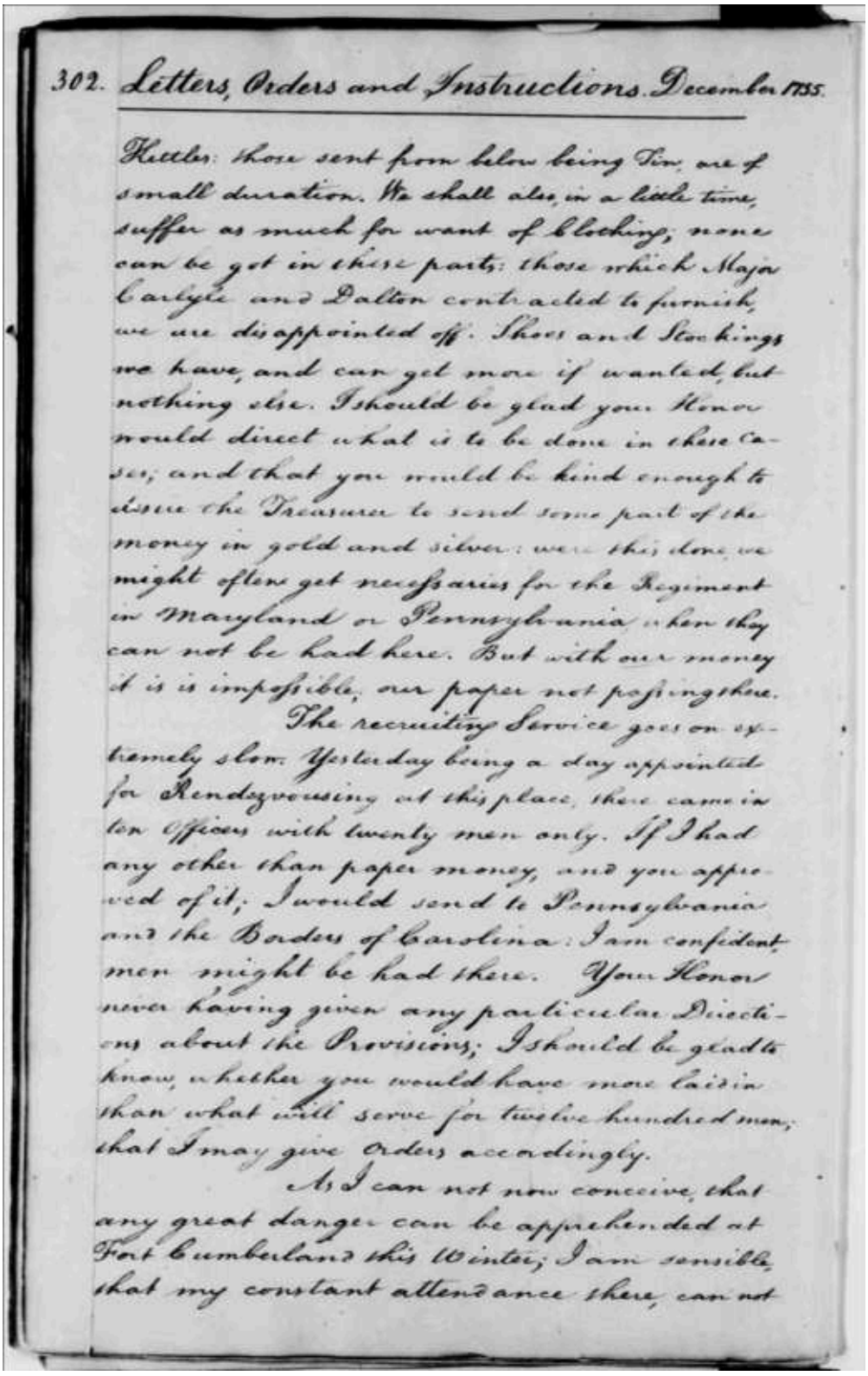
1 INTRODUCTION

Currently, there are many tried and tested techniques for automatically indexing and searching textual data, if the data is already in machine-readable form. That is, these techniques work only if the data is stored on disk in ASCII, Unicode, or a similar format. When the data is on a printed page, however, additional steps must be introduced before traditional information retrieval techniques may be applied. The pages must be somehow scanned and converted to ASCII or some other machine-readable format.

One method of creating such a machine-readable document from a collection of scanned manuscripts is to create a text index from the original documents. An index, as discussed here, is similar to the index in the back of a book: it would contain a list of words, each word associated with a set of links, one link for each occurrence of that word in the collection. With these indices, libraries can put their collections of manuscripts online, easily accessible to anyone who needed to do research involving these documents.

Traditionally, the conversion to ASCII format is done using Optical Character Recognition (OCR), which works well if the page is typewritten and there is little background noise present. When the page is handwritten, these methods fall apart. There is simply too much variation in how the same letter is written, even within a single author's collection, for OCR to be effective. Thus, a new system must be developed to create machine-readable representations of these handwritten pages, so collections of them can be indexed.

There are several applications for such an indexing system for handwritten pages. There exist large collections of handwritten documents, penned by the same author. Such bodies of work exist, for example, as part of the W. E. B. DuBois collection at the University of Massachusetts or the sets of presidential papers held at the Library of Congress. Certainly these documents are a very valuable resource, but leafing through



302. Letters, Orders and Instructions. December 1755.

Kettle: those sent from below being Tin, are of small duration. We shall also, in a little time, suffer as much for want of Blocking, none can be got in these parts: those which Majr Carlyle and Dalton contracted to furnish, we are disappointed off. Shoes and Stockings we have, and can get more if wanted, but nothing else. I should be glad your Honor would direct what is to be done in these Cases; and that you would be kind enough to desire the Treasurer to send some part of the money in gold and silver: were this done, we might often get necessaries for the Regiment in Maryland or Pennsylvania when they can not be had here. But with our money it is impossible, our paper not passing there.

The recruiting Service goes on extremely slow. Yesterday being a day appointed for Rendezvousing at this place, there came in ten Officers with twenty men only. If I had any other than paper money, and you approved of it, I would send to Pennsylvania and the Borders of Carolina: I am confident men might be had there. Your Honor never having given any particular Directions about the Provisions; I should be glad to know, whether you would have more laid in than what will serve for twelve hundred men, that I may give Orders accordingly.

As I can not now conceive, that any great danger can be apprehended at Fort Cumberland this Winter; I am sensible, that my constant attendance there, can not

Figure 1: One of George Washington's handwritten pages

hundreds or thousands of pages within these collections for specific information is a formidable task for even the most dedicated researcher.

A system that is able to automatically generate indices for these collections would help anyone who may want to use these resources. Manmatha and Croft (1998) suggested an algorithm, known as "word spotting," to create such an index from scanned pages of documents. Manmatha and Srimal (1999) developed an algorithm to segment pages into words used in the implementation of the word spotting algorithm presented here. The word spotting algorithm, first developed in (Manmatha and Croft 1998) is outlined as follows:

1. A scanned greylevel image of the document is obtained.

2. The page is divided into units (words) which can be compared singly.

3. Each word image is taken as a template and compared against all other word images.

4. Word classes are generated based on their similarity.

5. A human provides an ASCII equivalent of each class of similar words. (Manmatha and Croft 1998)

Step 1 is assumed to be done. The successful algorithm developed by Manmatha and Srimal (1999) handles Step 2. The remaining sections of the paper will present a comparative study of different matching techniques as an attempt to find a solution for Step 3. This investigation is more extensive than that in (Manmatha and Croft 1998) and is the first such attempt at applying these techniques to George Washington's manuscripts.

The work presented here is all based on a subset of Washington's manuscripts, scanned from microfilm and received from the Library of Congress. In his later years, Washington hired secretaries to edit and recopy some of his documents. Therefore the data set is a relatively homogeneous collection of documents made up of only a few people's handwriting.

2 BACKGROUND

Traditionally, text is segmented into lines, then words, then characters, and then Optical Character Recognition (OCR) techniques can be used to recognize the words. While traditional OCR works well for machine fonts (Manmatha and Croft 1998), the techniques are often ineffective on handwritten pages due to the variation in different instances of the same letter. As an alternative, the authors in (Marti and Bunke 2001) attempt to use a statistical model of language to avoid segmenting lines of text into words. They transform a row of handwritten text into a sequence of windows and then identify nine features for each window. Such features include the number of black pixels in the window, the centroid of the window, the second-order moments of the window, the upper and lower contours of the window, the gradient of the upper and lower contours of the window, the number of black-white transitions in the vertical direction, and the number of black pixels between the upper and lower contours. From this myriad of features, the authors create a hidden Markov model for each letter of the alphabet, and combine the characters into words and the words into sentences. The results are promising, but still rely on relatively separate letters, which is not always possible in general handwritten documents.

This problem may be avoided by segmenting the text into words and comparing entire word images against each other. For the work done here, the algorithm outlined in (Manmatha and Srimal 1999) is used to segment the document into words. This technique has been proven to work well (77-96 percent accuracy on Washington's

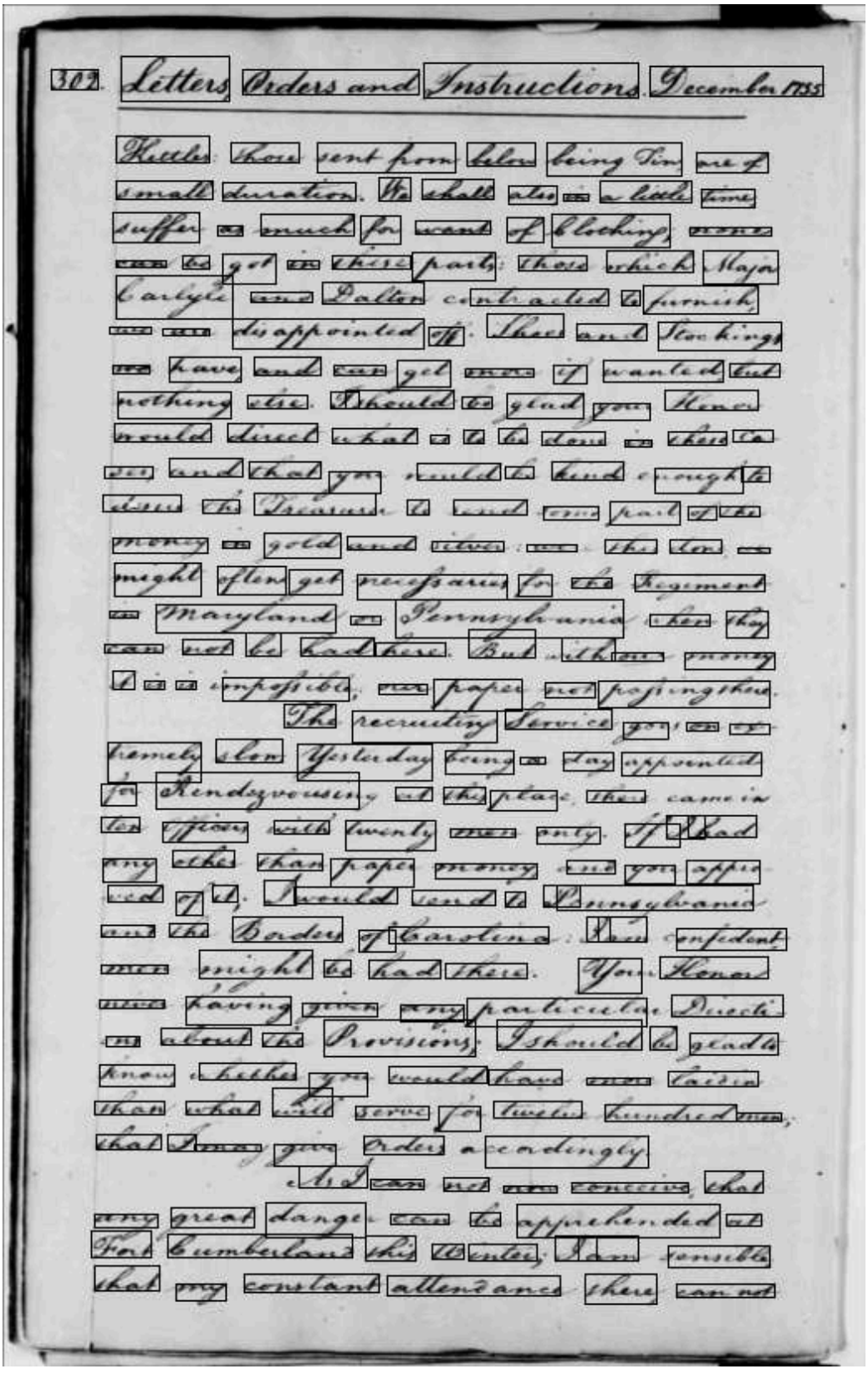


Figure 2: Page from Figure 1, after the segmentation process.

manuscripts), and was used as a preliminary step in the work presented here. Figure 1 shows a page of Washington's manuscripts and Figure 2 shows the page and the divisions made by the segmentation algorithm.

Once the text has been segmented, the task remains to sort these segments into classes. This will be done by comparing one segment, the template, against the others, and ranking matches by similarity scores. The top matches would then form the class. A preliminary study in (Manmatha and Croft 1998) identified two algorithms for matching images. The first is the Euclidean Distance Mapping (EDM) algorithm, which weights mismatches between words based on the magnitude of disparities. EDM can handle translations, but is unable to compensate for scaling, shear, or rotation – these transformations must be dealt with before the score is computed. The other method, an implementation of the feature association algorithm developed by Scott and Longuet-Higgins (1991), looks for an affine transformation between two images. Because these algorithms are very resource-intensive, it is desirable to prune the images which are most likely not to be matches, to cut down on the number of comparisons to be done by the matching algorithms.

3 OUTLINE OF MATCHING STEPS

With the page segmented into words, the next step is to match the words to each other to generate lists of similar images. The next three sections discuss the various steps to obtain these matchings:

1. The set of all images is pruned by discarding words which are unlikely to match the template.

2. The images are preprocessed to eliminate noise and normalize variations.

3. A score is generated for each comparison. High scores should indicate a good match and low scores should indicate a poor match.

To generate the scores for comparisons, four algorithms were tested. A simple algorithm to compute the XOR of two images was tested first. Next, an algorithm which calculated the Sum of Squared Differences (SSD) between two images was tested. The performance of the XOR algorithm was then improved through the use of Euclidean Distance Mapping (EDM). Finally, an algorithm based on the feature-matching algorithm suggested in (Scott and Longuet-Higgins 1991) was tested. All of these algorithms used one word as a template to compare against all the other words within the set. The details of these methods are explained in Section 6.

4 PRUNING

Pruning is the process of determining which images are not likely to be matches before running the scoring algorithm by using easy-to-measure properties of the image. Due to the resource-intensive nature of the matching algorithms, it is desirable to prune the search space of as many word images as possible, while being careful not to dismiss valid matches. Several types of pruning criteria were tested: image length, image area, aspect ratio, number of ascenders and descenders, and number of black-to-white transitions in a cross-section of the image.

The fastest pruning methods used depended strictly on the size of the image. Since there is a relatively low variation in the size of words written by the same individual, the length of a word image may be an estimate of the number of letters in the word. Thus it is initially assumed that there exists an α such that:

$$\frac{1}{\alpha} \leq \frac{template\_length}{image\_length} \leq \alpha.$$

Threshold α	Rel. Returned/ Relevant	Rel. Returned/ Returned	Returned (out of 2381)
4	1.000	0.012	2096
3	1.000	0.014	1847
2	0.982	0.019	1278
1.5	0.960	0.032	773
1.2	0.851	0.060	377
1.15	0.739	0.068	285

Table 1: Threshold data for α (length). Rel. Returned/Relevant is an accuracy rating at each threshold value, taken as the number of relevant words that were left after pruning at a given threshold value divided by the total number of relevant words. (Two word images are relevant if their ASCII translations are the same.) Rel. Returned/Returned is the number of relevant words left after pruning divided by the total number of words left after pruning. This score indicates how many non-relevant word images were left after pruning at each threshold. The final column, Returned, is simply the average number of word images left after pruning at each threshold. This table format is also used in Tables 2, 3, and 8-11.

Table 1 shows demonstrates how well pruning by length works at different values of α. It was observed that pruning by image length alone is likely to mismatch images that contain the same word written at different scales. Unfortunately the words in Washington's document do vary in size and this is sometimes magnified by the segmentation algorithm. To account for this, the authors in (Manmatha and Croft 1998) prune using the ratio of the areas of the word images and the ratio of the aspect ratios of the two words. That is, it is assumed:

$$\frac{1}{\beta} \le \frac{template\_area}{image\_area} \le \beta \text{ and}$$

$$\frac{1}{\delta} \le \frac{template\_aspect}{image\_aspect} \le \delta$$

$$\text{where } area = length * width \text{ and } aspect = \frac{width}{length}.$$

Threshold β	Rel. Returned/ Relevant	Rel. Returned/ Returned	Returned
4	0.996	0.014	1692
3	0.994	0.017	1404
2	0.939	0.024	936
1.5	0.863	0.037	554
1.2	0.644	0.054	258
1.15	0.473	0.052	190

Table 2: Threshold data for β (area).

Threshold δ	Rel. Returned/ Relevant	Rel. Returned/ Returned	Returned
4	1.000	0.012	2299
3	1.000	0.012	2160
2	1.000	0.015	1715
1.5	0.951	0.025	1158
1.2	0.737	0.039	580
1.15	0.661	0.044	431

Table 3: Threshold data for δ (aspect ratio).

This assumes that words written in different sizes will likely be scaled proportionally in both the vertical and horizontal directions and thus will have a similar aspect ratio to the template word. These two techniques together provided a better estimate than length alone for the pruning step, but still did not achieve the desired high level of pruning while not removing valid words from consideration.

To improve upon these techniques, the next pruning method attempted was to take into account features in the pixels of the words. One such pattern, an ascender, is the part of a letter which extends above the body of the word. Similarly, a descender extends below the main section of the word. For example, the word "example" has one ascender, the "l", and one descender, the "p", while the word "one" has neither ascenders nor descenders. To find the number of ascenders, first the upper and lower baselines of the main segment of the word were found. To compute the baselines of the image, the vertical projection of the black and white image is computed. The upper and lower baselines are taken as the two points where the changes in this projection are greatest. Figure 3 demonstrates this concept. Each connected component above the upper baseline was then counted as an ascender. Each connected component below the lower baseline was counted as a descender. (To avoid counting false ascenders and descenders, the components had to be larger than 10 pixels to be counted.) Tables 4 and 5 show the distribution of ascenders and descenders over all the words in the test set, and Tables 6 and 7 show the distribution of ascenders and descenders for a single word, Alexandria.

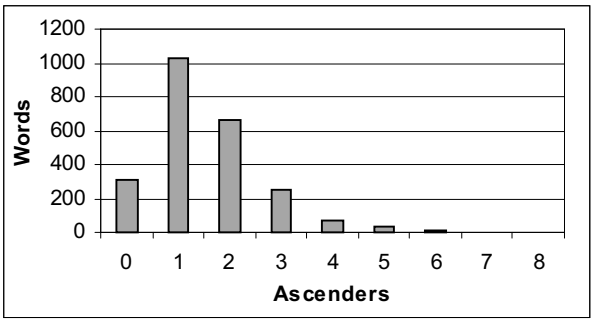


Table 4: Distributions of ascenders for all words in test set.

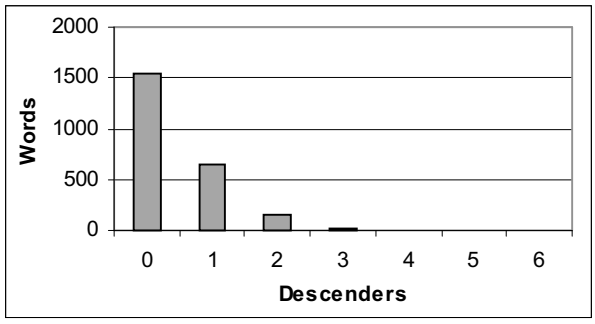


Table 5: Distributions of descenders for all words in test set.

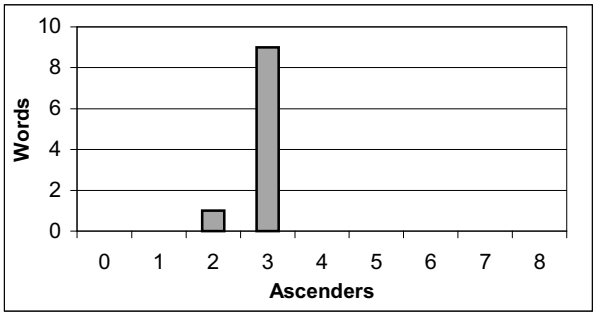


Table 6: Distribution of ascenders for one word, Alexandria.

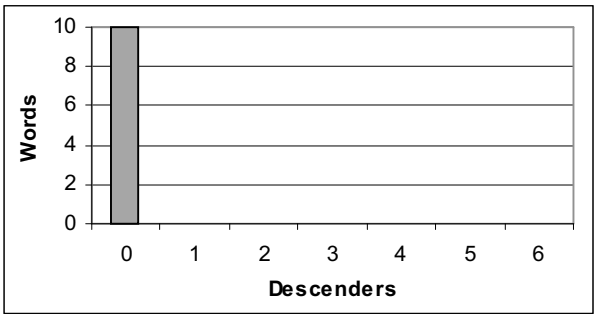


Table 7: Distribution of descenders for one word, Alexandria.

As expected, there is a small variation in the number of ascenders and descenders detected for one word class, but, unfortunately, the variation over all words is not large enough to be exploited, as seen in the above density plots. Counting ascenders was not always accurate, and thus pruning by ascenders alone often pruned out too many matches or allowed too many matches to be considered. Counting the number of descenders proved to be more accurate -- it was almost always the case that all the images for one word class would have the same number of descenders. Therefore pruning by the number of descenders was combined with other pruning techniques to improve the accuracy of the pruning step.

Because of the problem of accuracy in counting ascenders, another pruning method was used. This technique selected a cross-section of a binarized word and took a count of the number of black-to-white transitions within that cross-section. Using this technique, it was hoped that the number of letters in the word could be estimated more accurately than just by using the length of the image. Crosssections were considered to be a single row of the image, taken at the middle of the image, the centroid, the upper baseline, and the lower baseline. Figure 3 shows the location of each cross section on the word "December." Tables 8 through 11 show the number of documents returned for cross-section differences. The standard deviation of the number of transitions over all documents in the test set was approximately 5 (actual value 5.51); however, without preprocessing, the standard deviation of the number of transitions among non-stopwords varied by more than this. Because of this variation, the number of transitions within a crosssection did not yield accurate pruning results. Cutoff thresholds set in increments of one standard deviation allowed many non-relevant documents or too few relevant documents, as shown in the tables below.

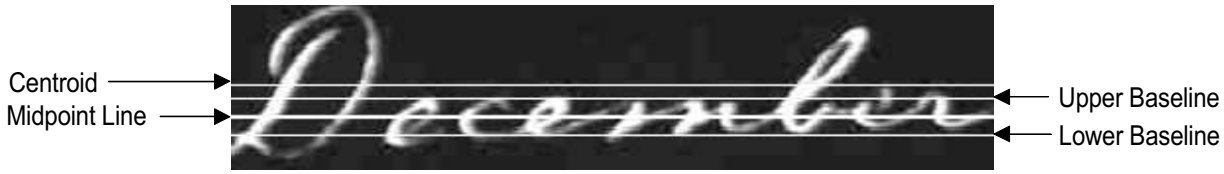


Figure 3: Diagram demonstrating the possible cross-section rows.

Threshold	Rel. Returned/ Relevant	Rel. Returned/ Returned	Returned
25	1.000	0.012	2375
20	1.000	0.012	2362
15	0.926	0.012	2318
10	0.808	0.012	2205
5	0.726	0.013	1903
0	0.014	0.001	71

Table 8: Threshold data for centroid transitions

Threshold	Rel. Returned/ Relevant	Rel. Returned/ Returned	Returned
25	1.000	0.012	2374
20	1.000	0.012	2358
15	0.968	0.012	2302
10	0.818	0.013	2174
5	0.710	0.013	1872
0	0.008	0.013	63

Table 9: Threshold data for middle transitions

Threshold	Rel. Returned/ Relevant	Rel. Returned/ Returned	Returned
25	1.000	0.011	2381
20	1.000	0.011	2380
15	1.000	0.012	2377
10	0.958	0.011	2357
5	0.892	0.012	2216
0	0.067	0.001	155

Table 10: Threshold data for upper baseline transitions

Threshold	Rel. Returned/ Relevant	Rel. Returned/ Returned	Returned
25	1.000	0.011	2380
20	1.000	0.011	2380
15	1.000	0.012	2377
10	1.000	0.012	2365
5	0.990	0.012	2299
0	0.113	0.029	252

Table 11: Threshold data for lower baseline transitions

Of the pruning techniques tried, a combination of examining area (with $\beta = 2$), aspect ratio (with $\delta = 1.5$), and number of descenders (only words with the same number of descenders are allowed) gave the best results. With these parameters, 87% of the possible comparisons were avoided, and 94% of the relevant words were retained. These techniques were used for the tests in the subsequent steps.

5 PREPROCESSING

After pruning out images that were least likely to match a given word image, some normalization of the images was needed so that the images could be better compared. These normalization steps were termed "preprocessing" in the sense that they were done before sending the images to the matching algorithms.

The first step was to binarize the images by converting all pixels less than a threshold value to white and all the others to black, to give white-on-black images, as in Figure 5 (the original is shown in Figure 4). Visual inspection over a set of images showed that a threshold value of 182 produced clear images with minimal loss of information. (A more robust thresholding scheme, such as examining the histogram of grey levels, may improve results.) Often a word to be matched had dangling descenders or other artifacts left by the segmentation algorithm from the words surrounding it in the original document. These artifacts negatively affected the matching algorithms and thus needed to be removed without removing parts of the actual word. To achieve this,

after the images were binarized, a white box was drawn between the upper and lower baselines and the 8 connected components of the resulting image were found. All the components but the largest were removed. This had the effect of removing the unwanted parts while keeping the ascenders intact with the original word. Figure 4 shows an image with a dangling descender and Figure 5 shows the results after binarization and component removal.

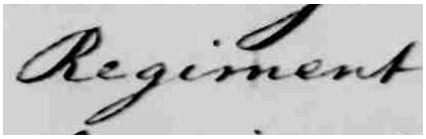


Figure 4: Original greyscale image of Regiment

Figure 5: Binarized version, with extra components removed.

After the removal of unwanted components, the next step was to compensate for the variation in the size of a word. The heights of the image and template were normalized so that the distance between the lower baseline and the top was the same for both words. The larger image was reduced and the smaller image was enlarged so that the two images were normalized to their average height.

After the heights of the words were normalized, the question was left of how to normalize the widths of the words. Several techniques were tried. First, nothing was done. The words were scaled according to their heights alone and any difference in width after the scaling was ignored. The results were promising, but not accurate enough. A second method was to first normalize only the heights of the words and then normalize the widths separately, and tests showed this to be an improvement. Even so, there were often cases where the first or last letter of a word had been not included due to the segmentation process, and this affected how well the words could be scaled to each other. In an attempt to compensate for this, the words were scaled by the distance between two centroids, one for the left half of the image and one for the right half, rather than overall width. Unfortunately, this technique was also prone to missing or cropped letters and thus did not improve results over the second method. Normalizing by relative height and then by width separately helped to remove any bias in the XOR computation (a step used in two of the matching algorithms as described in Section 6) due to the variation in sizes of the same word.

Another factor to be considered is that much of George Washington's writing is written at a slant, but, as with word size, this slant is not consistent throughout his papers. To correct for this, a slant angle was computed for each word and the image was sheared to remove this angle before computing the XOR image. The angle was computed by finding the average distance from the left edge of the image to the right edge of the first letter. The shear transformation was performed by shifting each row of the image by a distance proportional to the height of the row and the slant angle. This often yielded results similar to the transformation shown in Figure 6. Problems sometimes arose because of curls or tails on the first letter of a word; a more sophisticated method of finding the slant angle, such as calculating the slant angle at several points in the word image and averaging the results, may improve the slant correction process. In addition to a slant in the words, occasionally there was also a slight rotation present, but most of the words were horizontal. Because these occasional slight rotations did not greatly affect the XOR computations, the rotation was not normalized, but this is another possibility for improving the normalization process.

Figure 6: Regiment, after binarization and slant correction.

Other preprocessing methods varied based on which algorithm was to be used. Given a greyscale image, the edges of the letters can be found using a Gaussian derivative edge filter, as in Figure 7. A similar method involved applying a Laplacian of Gaussian filter to the image and thresholding the result using a value of 6 (determined empirically) to leave only the centers of the letters, as shown in Figure 8. These two techniques were

used to reduce the number of points the SLH algorithm needed to deal with, which helped to speed up computations. These techniques did not work for the EDM algorithm, however. Although the edge or center images are "cleaner" because most of the noise due to scanning and binarizing has been removed (for example, compare Figures 5 and 8), there are fewer pixels for the XOR to match and so errors in alignment are magnified. This explains the lack of improvement for the EDM algorithm with these techniques.

Figure 7: Edges extracted using a Guassian filter.

Figure 8: Extraction with a Laplacian of Guassian filter.

After the preprocessing steps were done, the images were aligned with each other in an attempt to generate the best matching possible between the two images. First the images were aligned vertically by their lower baselines. Three different methods were tried to align images horizontally. One method simply roughly aligned the images by their leftmost pixels. Another method attempted to align the centroid of the images. A final method attempted to line up the ascenders of each image. The method used most often in testing was aligning images horizontally by their leftmost pixels. The other two methods did not improve results and took more time to compute.

6 MATCHING ALGORITHMS

Four different matching algorithms were tried, ranging from a very simple XOR computation to a much more sophisticated calculation of affine transformations. All the matching algorithms took one image, designated it as a template, and took each of the other images left after the pruning step and compared them to the template to generate a score. In some way, all algorithms measured the error between the template and each image, thus a perfect match would have a score of zero. These scores were then inverted, and the images were ranked according to how well they matched the template.

XOR

The most basic of matching techniques implemented involved taking a simple XOR of two images. The images were aligned as discussed in the preprocessing stage. Then the XOR was computed of the corresponding pixels, which created a new binary image such that white pixels indicate mismatches between the two images. Figure 9 shows a demonstration of this process. These mismatched pixels are counted, and the result is used as the score for that set of images. This process is repeated a few times with different positioning of the two images for each set, allowing for shifts in the images (as the alignment step is imperfect). The minimum score from these comparisons. divided by the size of the XOR image was used as the match score for the two images. This technique was very fast, but was too general and did not fare well.

Figure 9: Two examples of "Alexandria," after preprocessing. On the right is the corresponding XOR image. The white pixels in the XOR image represent mismatched pixels.

SSD

The next attempt involved computing the Sum of Squared Differences (SSD) between two images. In computing the SSD, one of the test words is designated the template. The smaller of the two images was used as the template in the tests, though selecting the larger of the two images had no significant effect. The upper left pixel of the template image is then aligned with a pixel on the image to be tested. The SSD value for that pixel was computed using the formula

$$SSD(x,y) = \sum_{i}^{n}\sum_{j}^{m}\left[I_{(x+i)(y+j)} - T_{ij}\right]^2$$

where T is the $m \times n$ template image and I is the image to be tested against. This value was computed for every point in the test image. Once the SSD value had been computed for every pixel in the test image, the lowest of these values was then returned as the relevance score.

It was soon discovered that scores based on the SSD of two images were biased significantly toward pairs of images in which the template image was very small. When the template image is small, the SSD computation involves fewer values, resulting in a low error value (and an erroneously high score). This bias was adjusted for using two methods: a normalized version of the SSD algorithm of the form

$$SSD'(x,y) = \frac{SSD(x,y)}{\sqrt{\sum_{i}\sum_{j}\left(T_{ij}\right)^2}\sqrt{\sum_{i}\sum_{j}\left(I_{(x+i)(y+j)}\right)^2}}$$

and an additional preprocessing step in which the two words were scaled to similar sizes by the height of their letters (See Section 5 for more about size normalization).

This technique yielded results that were roughly equivalent to XOR-based image matching, though the time needed to compute the score was significantly higher. This time could be improved slightly by aligning the two images by their respective baselines and limiting the vertical movement of the template. With this additional constraint, the SSD tests ran significantly faster, with negligible loss of precision.

EDM

To improve on the XOR matching algorithm, it was noted that in the XOR image an isolated white pixel has the same value as a white pixel within a group of other white pixels. But an isolated white pixel is probably just due to noise left by the binarization process or a small misalignment in the images, whereas a blob of white pixels indicates a more serious mismatch between two images. Thus, to improve the score generated by the XOR algorithm, the pixels within a blob should be weighted higher than isolated pixels.

To achieve this desired weighting, Euclidean distance mapping (EDM) may be applied to the XOR image. This serves to weight the white pixels by their distance from black pixels. Thus a white pixel surrounded by other white pixels receives a higher weight than a white pixel surrounded by black pixels. To generate the weightings, a two-pass, low-error EDM variant called Sequential Octagonal Distance mapping (Danielsson 1980) was used. A score is then computed using the sum of the distances computed for each pixel in the XOR image. By using EDM, the rankings of words improved significantly, better than the two previous algorithms.

One improvement to the EDM scoring method is to attempt to normalize by the size of the images used in the comparison. The larger the word images, the more area for pixels to miss, resulting in worse EDM scores. Three techniques were tried to rid the scores of this bias. One method simply divided the score by the area of the XOR image. Another method that was tried divided the score by the original area of the smaller image, in an attempt to give better scores to images that were close in size. Unexpectedly, this second method did more poorly than the first. One possible explanation is that the sizes are already adjusted in the preprocessing steps, which means that dividing by the area of the XOR image does, in effect, weight scores lower for images that were not close in size. The final method of score normalization divided the score by the number of white pixels in the XOR image. This improved results, but not as much as dividing by the area of the XOR image, and also needed to be computed more often, so the first method was usedfor the tests.

SLH

Though the tests conducted using Euclidean distance mapping were encouraging, further precision is desired. To improve upon the results of the EDM tests, an algorithm suggested by Scott and Longuet-Higgins (1991), referred to here as the SLH algorithm, was implemented.

The SLH algorithm is intended to recover the correspondence between two sets of points *I* and *J*, when there exists an affine transform between them. An affine transform is a linear transformation of the form

$$\mathbf{r}' = \mathbf{A}\mathbf{r} + \mathbf{t}$$

where **A** is a *2 × 2* matrix and **t** is a 2-dimensional vector. This type of transform can describe scaling, shear, translation in both directions, and rotation between two sets of points.

In the SLH algorithm, the correspondence between points is calculated by first creating an adjacency matrix between all points $i \in I$ and all points $j \in J$. This matrix, called **G**, is of the form

$$G_{ij} = e^{\frac{-r_{ij}^2}{2\sigma^2}}$$

for each point in *I* and *J*. (r_{ij} is the Euclidean distance between points *i* and *j*.) This matrix **G** is diagonalized to **G = TDU** using singular value decomposition (SVD), resulting in the diagonal matrix **D** and orthogonal matrices **T** and **U**. The matrix **D** is then thresholded to find the matrix **E**: all corresponding diagonal values of **D** greater than γ are set to 1 in **E**, and all other values of **E** are set to 0. The matrix **E** is then used to compute **P = TEU**, where **P** is a pairing matrix describing the correspondence between points in *I* and *J*. More specifically, a correspondence between points *I* and *J* is suggested when P_{ij} is the greatest value both in its row and column. Once this correspondence is found, the appropriate affine transform can be easily calculated using the least mean squares method.

This algorithm may be utilized to determine feature correspondence between a pair of images. First, the images are thresholded (as discussed in Section 5), and a collection of points (all non-zero pixels) is derived from each resulting binary image. These collections of points *I* and *J* are then passed to the SLH algorithm. Once the correspondence is found between the points of *I* and *J*, the error between the transformed set of points *J'* and *I* (ESLH) may be computed as follows

$$ESLH = \sum_{v} \left(I_v - \mathbf{A}J_v - \mathbf{t} \right)^2$$

where *v* represents a single point in *I* or *J*. This error score is smallest when the images being compared are most similar, and thus its inverted value may be used as a scoring function.

Prior research using this method has provided encouraging results (Manmatha and Croft 1998). However, while this earlier work used SLH on a set of images pruned by their EDM score, the SLH algorithm was used here as a distinct image matching function.

Results for the SLH algorithm were encouraging, though the results found using SLH alone were less accurate than those found using the EDM test. The effectiveness of the algorithm in matching words was affected by several factors, including the values of σ and γ. Empirical testing showed that a values of 2.0 for σ and of 10^{-6} for γ were ideal for these tests.

The techniques used to generate the binary images needed for the SLH algorithm also had a significant impact on the effectiveness of this method. Thresholding the images at varying grey levels was attempted, as was edge detection and skeletonizing in various ways. The images acquired using the thresholded image after applying a Laplacian of Gaussian filter (as in Figure 8) gave the best results.

While the experiments using SLH as a matching algorithm were encouraging, the time needed by the algorithm was far greater than was needed for any of the other algorithms tested (Table 15). This is primarily due to the time necessary to perform singular value decomposition on a large matrix; faster methods to compute the SVD of a large matrix would shorten the running time of this algorithm considerably.

7 EXPERIMENTAL RESULTS

For a set of test data on which to test the algorithms, three sets of 10 consecutive pages were chosen from George Washington's letters. These letters are part of the archival collection at the Library of Congress, where they have been scanned in and kept on microfiche. The images of the documents used in the experiments were scanned from the microfiche. Therefore, because these were third generation copies, there was significant degradation in the quality of the image; unwanted artifacts had been introduced and image quality had been lost.

With each of the three sets of data taken from the collection, the pages were segmented into word images using the techniques described in (Manmatha and Srimal 1999). In order to create a ground truth to score how well the matching algorithms did, it was then necessary to manually generate a listing of the ASCII equivalent for each word. This was a tedious and difficult process, due to the poor quality of the images. It quickly became apparent how useful a system to automatically index these documents would be. Because the segmentation algorithm was not perfectly accurate, some of the words had been cropped. If less than a letter was missing, the entire word was given as the ASCII equivalent. If more than that was missing, however, only the part of the word that was included was given as the ASCII equivalent. If no letters could be made out, a control character ('#') was inserted into the translation.

By using this transcription of the images, relevance between images could be judged. An image was deemed relevant to another if the two images had the same ASCII equivalent. When a ranked list of matches was returned by one of the matching algorithms, recall-precision scores were computed.

After running tests on the three data sets, it became apparent that two of the sets of 10 pages were of extremely low quality and did not allow any of the matching algorithms to work well. These pages also caused the most segmentation errors and difficulty in transcription. Because the quality of the images within these two sets were so poor, all the tests for comparing algorithms were run on the third set, which posed fewer problems. In addition, the third data set contained the most repeated non-stopwords that would be useful to index and could be chosen for tests. Table 12 shows the differences in average recall-precision scores between the three sets. Figure 10 shows three lines, one from each data set.

	Data Set 1	**Data Set 2**	**Data Set 3**
Average Precision	0.4956	0.3365	0.7338
R-Precision	0.4852	0.3319	0.6846

Table 12: Comparison of recall-precision scores (using EDM for a matching algorithm) across the three data sets.

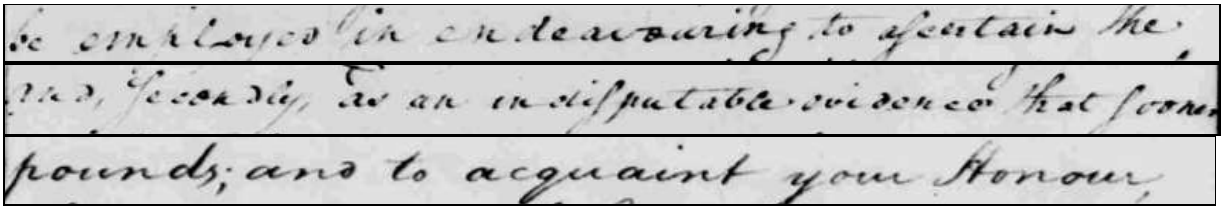


Figure 10: Representative lines extracted from data sets 1, 2, and 3, respectively. Note the uniformity in ink in the third line as compared with the other two. The difference in quality is more pronounced when the images are displayed on a screen, and also magnified when the images are binarized.

To create a test set to test the algorithms, 15 words were chosen of varying lengths (2 to 14 characters) and varying number of occurrences (4 to 110 appearances). Each of these words were run through the matching algorithm, to generate a ranked list, and the average recall-precision scores were calculated over all 15 queries. Table 13 shows the differences in how the algorithms performed over these queries. Table 14 shows the differences in EDM performance when preprocessing was used. Following the tests to determine average recall-precision, the running times of the different algorithms were tested; Table 15 shows the time required to perform a single comparison using each algorithm.

	XOR	**SSD**	**EDM**	**SLH**
Average Precision	0.5414	0.5266	0.7338	0.4243
R-Precision	0.5011	0.4706	0.6846	0.3846

Table 13: Comparison of recall-precision scores for the four matching algorithms.

	Without Preprocessing	**With Preprocessing**
Average Precision	0.0832	0.7338
R-Precision	0.0645	0.6846

Table 14: Comparison of recall-precision scores for EDM with and without preprocessing.

	XOR	**SSD**	**EDM**	**SLH (half)**	**SLH (full)**
Running Time (seconds)	13.0	72.0	14.3	121.4	5508.8

Table 15: Comparison of running time for the four matching algorithms. The scores here indicate the time required to perform a single comparison between two instances of the word Alexandria. For SLH, the tests were run on both the half-size and full-size images.

The results found in the experiments indicate that the EDM algorithm is roughly as fast as XOR, but more importantly is the most accurate matching method of those tested, with recall-precision scores well above those achieved using SLH. However, due to the enormous amount of time required to compute the proper affine transform, the images used by the SLH algorithm were first reduced in size by half in order to reduce the number of points involved and improve performance. The exact effect of this reduction on the performance of the algorithm has not yet been determined. It is believed that performance may increase considerably when higher resolution images are used. Tests using SLH on full-size images were run, but were not able to be completed by the time of this publication. Further methods to improve the performance of the SLH algorithm have been considered, and are discussed briefly in Section 8. Figure 11 shows the top eight matches for three of the test words, as ranked by the EDM algorithm.

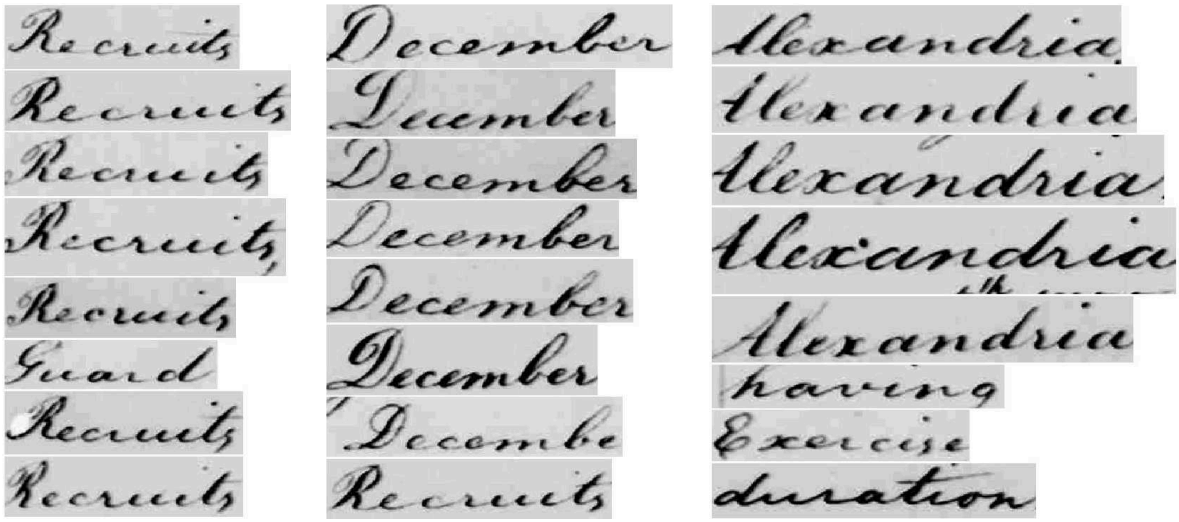


Figure 11: Top 8 matches from rankings, as determined by the EDM algorithm. The template is the top ranked image in each list (Recruits, December, and Alexandria, respectively). The average precision values for these three queries are 0.9526, 0.7764, and 0.6069, respectively. Similarly, the R-precision values are 0.8750, 0.6364, and 0.6000, respectively.

A few insights into improvements in the EDM algorithm can be gathered by viewing the results in the ranked lists presented in Figure 11. One observation is that the last entry in the list for "December" is actually the same word image as the second entry in the "Recruits" list. Thus, in an actual implementation of the matching algorithms, where word classes were to be formed, the "Recruits" ranked eighth in the "December" list would not be included in the "December" word class, as it appears more highly ranked in another list. By gathering these ranked lists into actual word classes, the recall-precision scores would be improved. Another observation to make is that the sixth, seventh, and eighth entries in the ranked list for "Alexandria" are much smaller than the "Alexandria" template. This suggests that if some sort of weighting was applied to the scores returned from EDM based on the size difference in word images, the rankings may be more accurate. These observations suggest some possibilities to improve the matching algorithms, several other possibilities are discussed in Section 8.

8 FUTURE WORK

Several directions have been proposed as part of work that could be continued in this area. They fall into three categories: better methods of pruning and preprocessing, additional testing, and implementing other algorithms.

The pruning and preprocessing steps are far from polished. Pruning is a difficult problem because it is desirable to narrow the search space for possible matches, but relevant words should not be missed. In addition, as more possible matches are considered, correct matches are more likely not to be ranked highly. Thus, it is desirable to narrow the search space as tightly as possible without missing relevant matches. Several techniques have been discussed and tried, but other methods may work better. Perhaps the best method is to not prune at all, but rather to try a simple, fast algorithm on all the images and then use more sophisticated methods on the top subset of matches returned by the first.

Additional preprocessing methods could also be implemented. Although most of Washington's handwriting was more or less horizontal, scores could be improved by attempting to normalize any rotation that might be present. When converting images from greyscale to black and white, a constant value was chosen as the

threshold. A more sophisticated method of thresholding, such as choosing a threshold based on a histogram of the original image may improve the quality of the black and white images.

An obvious weakness in this examination of matching techniques is the lack of test data. Due to the difficulty in manually indexing the documents for testing, the algorithms were only tested using a few pages from a large corpus of George Washington's handwritten documents. Additionally, the methods were not tried on any other set of handwritten manuscripts. But that is not to say that the results presented here would not generalize to other authors' handwriting. Indeed, the EDM and SLH algorithms have been tried on other sets of documents, including test cases from the DIMUND document server and the Hudson archival collection (Manmatha and Croft 1998). Regardless, before the methods presented here are used in an full indexing system, they would need to be tested across many different authors' documents.

There are other techniques that may prove to be of use in dealing with handwritten documents. One would be to model the word matching problem as a bipartite matching problem, or analogous network flow problem. Similar techniques have been used in image feature mapping (Cheng et al 1996). There are several other point-matching techniques that may be applicable, such as those discussed in (Gottesfeld Brown 1992). In addition, other weighting techniques could help to improve recognition scores. Techniques such as using hidden Markov models, while not accurate enough to produce match rankings, may help to weight words may prove to give more context to words and improve rankings (Marti and Bunke 2001). Another possible scoring method could use the feature comparison techniques that were used in pruning. Several features of an image could be computed, such as number and locations of ascenders/descenders, centroids, moments, areas, edge crossings, horizontal and vertical projections, moments; and these features could be combined to create a signature for that word. In comparing two words, a vector of such features could be computed for each image and then the two (possibly weighted) vectors could be compared to yield a score. Possibly the best methods could be derived by combining these or other techniques to create hybrid matching algorithms.

Finally, it is believed that the scores achieved by the SLH algorithm on the current test data are less than optimal. Several changes to the current implementation have the potential to improve performance. Using the higher resolution images currently used by the EDM algorithm would almost certainly improve performance. Reducing the number of points in the image using a technique such as skeletonizing may allow the higher resolution images to be used, while still keeping the running time at an acceptable level. Other methods for computing the score have also been discussed. Aligning the baselines of the two images and assuming the translation on the y-axis to be zero may produce a more accurate affine transform between the two images, though this has yet to be tested. The current implementation of the algorithm also does not factor the number of pixels in each image which are not given a corresponding point by the SLH algorithm. It has been observed that the number of these pixels is much smaller between words that are the same. It is believed that factoring the number of unmatched points into the SLH score may further improve accuracy of the algorithm.

9 CONCLUSION

There are two problems central to indexing large collections of handwritten documents. The first obstacle is segmenting a handwritten page into words that can be matched against other words. This has been largely solved by the scale-space technique described in (Manmatha and Srimal 1999). The second problem is the matching of the word images and classification into classes so that an index can be created. Though the study of techniques described here is not conclusive, it does give promising results and point to directions that may eventually lead to a viable indexing system. In particular, advancements in image normalization for the EDM algorithm or improvements to the running time of the SLH algorithm would create an accurate matching technique that could be further tested and eventually incorporated into an automatic indexing system for handwritten manuscripts.

10 ACKNOWLEDGEMENTS

This material is based on work supported in part by the Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623 and in part by the National Science Foundation under grant numbers EIA-9820309 and IIS-9909073. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

R. Manmatha provided guidance throughout our time spent on this project. FangFang Feng provided the sets of documents used in our experiments and the software to segment those documents into words.

11 REFERENCES

Cheng, Y., Wu, V., Collins, R., Hanson, A., Riseman. 1996. Maximum-Weight Bipartite Matching Technique and its Application in Image Feature Matching. *Proceedings of SPIE Conference on Visual Communication and Image Processing*.

Danielsson, Per-Erik, 1980. Euclidean Distance Mapping. *Computer Graphics and Image Processing.* 14: 227-248.

Gottesfeld Brown, L. 1992. A Survey of Image Registration Techniques. *ACM Computing Surveys*. Vol. 24, No. 4: 325-376.

Manmatha, R.. and Croft, W. B. 1998. Word Spotting: Indexing Handwritten Manuscripts. *Intelligent Multimedia Information Retrieval.*

Manmatha, R. and Srimal, N. 1999. Scale Space Technique for Word Segmentation in Handwritten Documents. *LNCS: Scale-Space Theories in Computer Vision*: 22-33.

Marti, U.V. and Bunke, H. 2001. Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System. *International Journal of Pattern Recognition and Artificial Intelligence*. Vol 15, No. 1: 65-90.

Scott, G. L. and Longuet-Higgins, H. C. 1991. An Algorithm for Associating the Features of Two Patterns. *Proceedings of the Royal Society of London B* B224: 21-26.